Math 156: Problem Set 2

Owen Jones

April 28, 2024

1. $\operatorname*{argmax}_{\mathbf{w}} P(\mathbf{w}|D,\alpha) = \operatorname*{argmax}_{\mathbf{w}} P(D|\mathbf{w})P(\mathbf{w}|\alpha)$
$= \operatorname*{argmax}_{\mathbf{w}} (\prod_{n=1}^{N} \frac{\sqrt{\beta}}{\sqrt{2\pi}} e^{-\frac{\beta}{2}(t_n - y(x_n,\mathbf{w}))^2})(\frac{\alpha}{2\pi})^{\frac{(M+1)}{2}} e^{-\frac{\alpha}{2}\mathbf{w}^\top \mathbf{w}}$
$= \operatorname*{argmax}_{\mathbf{w}} \log((\prod_{n=1}^{N} \frac{\sqrt{\beta}}{\sqrt{2\pi}} e^{-\frac{\beta}{2}(t_n - y(x_n,\mathbf{w}))^2})(\frac{\alpha}{2\pi})^{\frac{(M+1)}{2}} e^{-\frac{\alpha}{2}\mathbf{w}^\top \mathbf{w}})$
$= \operatorname*{argmax}_{\mathbf{w}} \frac{N}{2}\log(\frac{\beta}{2\pi}) + \frac{M+1}{2}\log(\frac{\alpha}{2\pi}) - \frac{\beta}{2}\sum_{n=1}^{N}(t_n - y(x_n,\mathbf{w}))^2 - \frac{\alpha}{2}\mathbf{w}^\top\mathbf{w}$
$= \operatorname*{argmax}_{\mathbf{w}} -\frac{\beta}{2}\sum_{n=1}^{N}(t_n - y(x_n,\mathbf{w}))^2 - \frac{\alpha}{2}\mathbf{w}^\top\mathbf{w}$
$= \operatorname*{argmin}_{\mathbf{w}} \frac{\beta}{2}\sum_{n=1}^{N}(t_n - y(x_n,\mathbf{w}))^2 + \frac{\alpha}{2}\mathbf{w}^\top\mathbf{w}$
which is the minimizer of our desired function.
The MAP approach adds a regularizer term to MLE approach.

2. Suppose for the sake of contradiction $\{\mathbf{x}_n\}$ and $\{\mathbf{y}_n\}$ are linearly separable, but there exists a point $\mathbf{z}$ where the two convex hulls intersect.
Let $\mathbf{z} = \sum_n \alpha_n \mathbf{x}_n = \sum_n \beta_n \mathbf{y}_n$.
It follows $\mathbf{w}^\top\mathbf{z} + w_0 = \mathbf{w}^\top(\sum_n \alpha_n\mathbf{x}_n) + w_0 = \mathbf{w}^\top(\sum_n \beta_n\mathbf{y}_n) + w_0$
$\mathbf{w}^\top(\sum_n \alpha_n\mathbf{x}_n) + w_0 = \sum_n \alpha_n(\mathbf{w}^\top\mathbf{x}_n) + w_0$ because α_n is a scalar quantity.
Using $\sum_n \alpha_n = 1$, we obtain $\sum_n \alpha_n(\mathbf{w}^\top\mathbf{x}_n) + w_0 = \sum_n \alpha_n(\mathbf{w}^\top\mathbf{x}_n + w_0)$.
Thus, we have $\sum_n \alpha_n(\mathbf{w}^\top\mathbf{x}_n + w_0) = \sum_n \beta_n(\mathbf{w}^\top\mathbf{y}_n + w_0)$. However, this is impossible for all $\alpha_n, \beta_n \geq 0$ because by assumption $\mathbf{w}^\top\mathbf{x}_n + w_0 > 0, \mathbf{w}^\top\mathbf{y}_n + w_0 < 0$. Hence, $\{\mathbf{x}_n\}$ and $\{\mathbf{y}_n\}$ are not linearly separable.

   Suppose for the sake of contradiction there exists a point $\mathbf{z}$ where the two convex hulls intersect, but the convex hulls are linearly separable.

It follows there exists a vector $\vec{\mathbf{w}}$ and scalar w_0 s.t $\mathbf{w}^\top \mathbf{x}_n + w_0 > 0$ and $\mathbf{w}^\top \mathbf{y}_n + w_0 < 0$. Since $\mathbf{z}$ lies on the intersection of the two convex hulls, $\mathbf{z} = \sum_n \alpha_n \mathbf{x}_n = \sum_n \beta_n \mathbf{y}_n$. However, $\sum_n \alpha_n(\mathbf{w}^\top \mathbf{x}_n + w_0) = \sum_n \mathbf{w}^\top(\alpha_n \mathbf{x}_n) + w_0 = \sum_n \mathbf{w}^\top \mathbf{z} + w_0 > 0$ and $\sum_n \beta_n(\mathbf{w}^\top \mathbf{y}_n + w_0) = \sum_n \mathbf{w}^\top(\beta_n \mathbf{y}_n) + w_0 = \sum_n \mathbf{w}^\top \mathbf{z} + w_0 < 0$ which is clearly a contradiction. Hence, the two convex hulls do not intersect.

3. See the function SGD in the attached code

4. For the purpose of this problem, I chose to use sklearn's MinMax scaler on the features in the dataset. The parameters I chose for my SGD classification model were to RSME less than 0.1 as the termination conditon, $100,000$ max iterations, 10% of total population as the batch size, and a learning rate of 0.005. I used a train, validation, test split of 60%, 20%, and 20%. The model performed very wel on the test set (See classification report). Precision, recall, and f-1 score were all in the low to mid $90s$ except for the the recall on class 0. Thus, the ration of True negatives to total number of negatives is slightly worse than the other metrics.

In [62]:

```python
import numpy as np
import sklearn as skl
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import classification_report
import pandas as pd
from numpy import linalg
from scipy.special import expit
```

In [3]:

```python
data=datasets.load_breast_cancer()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target
```

In [58]:

```python
def SGD(data,validate,epsilon=0.1,target=None,batch_size=None,alpha=0.005,max_it
    if batch_size is None:
        batch_size=int(0.1*data.shape[0])
        #sets batch size to 5% of population if none entered
    w=np.random.normal(0,1,data.shape[1])
    #generates initial weights and biases for model from standard normal
    for i in np.arange(max_iter):
        sample=data.sample(n=batch_size)
        #samples data from population
        x=np.insert(sample.iloc[:,:-1].values,0,np.ones([batch_size,]),axis=1)
        #adds column of ones for w_0 of weights and bias vector
        t=sample.iloc[:,-1].values
        #vector of targets
        y=expit(x.dot(w))
        #compute the y_n for each x_n
        g=np.array([diff*phi for diff,phi in zip(y-t,x)]).sum(axis=0)
        #compute the gradient
        w=w-alpha*g
        #compute new weight and bias vector
        if i%1000==0:
            x_v=np.insert(validate.iloc[:,:-1].values,0,np.ones([validate.shape[
            y_v=expit(x_v.dot(w))
            t_v=validate.iloc[:,-1].values
            error=np.sqrt(((y_v - t_v) ** 2).mean())
            if error<epsilon:
                break
    return w,error,i #return w vector, RMSE, and iterations
```

In [31]:

```python
scaler=MinMaxScaler()
scaled_data = scaler.fit_transform(df)
scaled_df = pd.DataFrame(scaled_data, columns=df.columns)
```

In [76]:

```python
scaled_df.head()
```

Out[76]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	syn
0	0.521037	0.022658	0.545989	0.363733	0.593753	0.792037	0.703140	0.731113	0.6
1	0.643144	0.272574	0.615783	0.501591	0.289880	0.181768	0.203608	0.348757	0.:
2	0.601496	0.390260	0.595743	0.449417	0.514309	0.431017	0.462512	0.635686	0.5
3	0.210090	0.360839	0.233501	0.102906	0.811321	0.811361	0.565604	0.522863	0.:
4	0.629893	0.156578	0.630986	0.489290	0.430351	0.347893	0.463918	0.518390	0.:

5 rows × 31 columns

In [38]:

```
train, validate, test = np.split(scaled_df.sample(frac=1), [int(.6*len(scaled_df
```

In [61]:

```
test_features=np.insert(test.iloc[:,:-1].values,0,np.ones([test.shape[0],]),axis
test_target=test.iloc[:,-1].values
predicted_target=expit(test_features.dot(SGD(train,validate)[0]))
```

In [75]:

```
print(classification_report(test_target,np.round(predicted_target)))
```

```
              precision    recall  f1-score   support

         0.0       0.93      0.88      0.91        49
         1.0       0.91      0.95      0.93        65

    accuracy                           0.92       114
   macro avg       0.92      0.92      0.92       114
weighted avg       0.92      0.92      0.92       114
```